Received May 7, 2015, accepted May 21, 2015. Date of publication June 8, 2015, date of current version June 22, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2442613

Open and Low-Cost Virtual and Remote Labs on Control Engineering

JACOBO SÁENZ[1], JESÚS CHACÓN[1], LUIS DE LA TORRE[1], ANTONIO VISIOLI[2], (Senior Member, IEEE), AND SEBASTIÁN DORMIDO[1]

[1]Department of Computer Science and Automatic Control, School of Computer Science, National University of Distance Education, Madrid 28040, Spain
[2]Department of Mechanical and Industrial Engineering, University of Brescia, Brescia 25121, Italy

Corresponding author: J. Sáenz (jacobo.saenz@bec.uned.es)

This work was supported by the Spanish Ministry of Economy and Competitiveness under Project DPI2012-31303.

ABSTRACT This paper presents an open course in the University Network of Interactive Laboratories, which offers several virtual and remote laboratories on automatic control, accessible to anyone. All the details on one of these labs (a two electric coupled drives system that allows performing control practices in a 2×2 MIMO system with industrial applications) and the activities that can be performed with it are given. We use a low-cost solution for developing the virtual and remote labs shared in this open course, based on the use of a free authoring tool Easy Java/Javascript Simulations (EJsS) for building the laboratories' user interfaces and a cheap development platform board (BeagleBone Black). The virtual and remote labs are deployed into a free Learning Management System (Moodle) Web environment that facilitates their management and maintenance.

INDEX TERMS Distance education, remote laboratories, control engineering education, control systems.

I. INTRODUCTION

Traditional face-to-face education can be complemented with online experimental frameworks. Fortunately, there are lots of free Internet resources to help with many aspects of education already. In this regard, engineering and scientific studies need specific Internet-based tools to complement the practical, hands-on, part of their teaching. However, some of these tools are not so commonly available openly, in such a way anyone could use them.

Learning Management Systems (LMSs) and web-based labs have become widespread in distance education in the last years. On the one hand, LMSs support the administration, documentation, tracking, and reporting of training programs, classroom and online events [1]. LMSs are the pillars that sustain a good amount of online educational resources and programs for all kind of studies. On the other hand, web-based labs have some advantages for both, students and teachers [2]. One of them is the possibility of illustrating scientific phenomena that require costly or difficult-to-assemble equipment [3]. These tools, are essential for complementing many courses with online material. Considering the distance education paradigm in scientific and technical areas, the use of online laboratories is essential in the learning [4] and research [5] processes. Web-based labs should ideally consist of two different and complementary parts:

- Virtual Labs (VLs) provide computer based simulations which offer similar views and ways of work to their traditional counterparts [6]. Nowadays, simulations have evolved into interactive graphical user interfaces where students can manipulate the experiment parameters and, explore their evolution. This allows students to study different systems and with basic programming skills, change the mathematical model or construct a new one [7].
- Remote Labs (RLs) use real plants and physical devices which are operated at distance [8]. Remote experimentation through the Internet has been available from nearly two decades [9], and its interest has not diminished over the years [10], [11]. Another indicator of nowadays interest in this field is the number of works published in this field (recent examples of this are [12]–[21]). Moreover, the appearance of mobile devices and new tactile actuators [22] are boosting their interest and applications even more.

Since VLs are nothing but computer applications, it is easy and cheap for their developers to share them with the community. This is the case of the ComPADRE-Open Source Physics [23] repository,[1] with hundreds and hundreds of Java and Javascript simulations, all of them created with EJS, or

[1]http://www.compadre.org/osp

2169-3536 © 2015 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

the PhET [24] webpage,[2] also with hundreds of simulations based on Java, Flash and HTML5.

RLs, however, require real equipment to run. This means that RLs are: 1) more expensive to create and maintain and 2) the maintenance to ensure their correct operation requires time and attention. Due to these two factors, it is rare to find open RLs on the web, freely accessible to anyone. This work helps with both problems by: 1) using low-cost hardware for monitoring and controlling the devices used by the RLs and 2) embedding them into a LMS for making their deployment, usage tracking and maintenance easy.

Even though LMSs and virtual and/or remote labs (VRLs) are complementary tools, there is still a lack of convergence and interoperability between both [25], although some works as [26] have helped with this issue. The present work applies a solution [27], that also solves this gap, to a few control VRLs which are offered online in an open course in the University Network of Interactive Laboratories (UNILabs)[3] web portal, supported by a LMS.

In terms of hardware, it is frequent that each existing RL is based on different equipment. This work applies a low-cost solution (BeagleBone Black boards[4]) for standardizing the way in which the previous equipment are monitored and controlled. In terms of software, it is also usual to find RLs applications that were built and deployed using different technologies. However, in UNILabs, all of them share the same solution, which is based on the use of three free and open source applications: Moodle[5] (the LMS that supports the web content of the open course in which the VRLs are), Easy Java Simulations or EJS[6] (for building the VRLs applications), and NodeJsonRpcElement[7] (a middleware created by the authors of this work in order to allow the communication between EJS and the BeagleBone Black board).

The BeagleBone Black board is a low-cost, open hardware, and community-supported embedded computer for developers. This board replaces the server PC in the architecture of RLs UNILabs has been using up to now. It has two important advantages: it is not as expensive as a PC with a DAQ system and it requires much less physical space in the laboratory. Moodle is a free, open source, and widespread used LMS (more than 60 million registered users, according to its official webpage) that supports constructivist learning, offering its users communication and interaction facilities. Finally, EJS [28] is also a free and open source tool, especially designed for the creation of discrete computer simulations. Recent releases of EJS support connections with external applications, such as LabView and Matlab/Simulink as well as with hardware, such as Arduino, Phidgets and, now, thank to this work, BeagleBone boards. Hence, EJS not only is useful for creating virtual labs, but also the GUIs of their remote counterparts [29].

The offer of VRLs in UNILabs open course includes: a DC motor (a handmade assembly), a heat flow system,[8] and a two coupled electric drives system.[9] While the DC motor and the heat flow system VRLs have already been presented in previous works [30], [31] and already were part of UNILabs, the RLs were only accessible to students from any of the universities that form part of the UNILabs network. There was only one DC motor and one heat flow system available to experiment with. However, the authors had three more DC motor assemblies and an additional heat flow system, none of which was being used. All these apparatus are now in use and offered as RLs, thanks to the low-cost solution presented in this paper. Readers who are interested in these resources are encouraged to visit UNILabs and enter in its open course. Finally, the two coupled electric drives system is a completely new VRL in the network. Therefore, this one is presented in detail and it is used as an example of the connection between EJS and the BeagleBone Black board.

The paper is organized as follows: Section 2 describes the new web-based control laboratory in UNILabs (the two coupled electric drives system), the physical plant and its mathematical model. Section 3 presents the available set of experiments for this new lab. Section 4 offers a view of UNILabs, the Moodle web portal into which the web-labs of the presented open course have been integrated. Finally, some conclusions are given in Section 5.

II. THE TWO COUPLED ELECTRIC DRIVES SYSTEM WEB-LAB

A. THE PLANT

The TecQuipment coupled drives apparatus is a compact self-contained bench mounting apparatus designed to allow students at all academic levels to investigate basic and advanced principles of control, including control of multi-variable systems. This plant shows the problems of controlling speed and tension in coupled drives. The presented plant is important in industry since many applications use coupled drives; as, for example: magnetic tape drives, textile machines or paper mills. This apparatus has two electric motors, coupled by a continuous flexible belt. The belt also passes over a swinging arm with a jockey wheel that measures the belt speed and tension. A manual control allows the user to adjust the spring tension at the swinging arm. The basic control problem is to vary the torque in the motors to regulate the belt speed and tension. Since there are two motors, the system has two inputs and the two controlled variables (speed and control) offer two outputs. Therefore, this is a 2×2 MIMO system.

[2]https://phet.colorado.edu/
[3]http://unilabs.dia.uned.es
[4]http://beagleboard.org/black
[5]https://moodle.org
[6]http://www.um.es/fem/EjsWiki/
[7]https://github.com/jcsombria/ejs-rhc

[8]Commercial assembly provided by Quanser http://www.quanser.com/Products/heatflow
[9]Commercial assembly provided by TecQuipment http://www.tecquipment.com/Control/Control-Engineering/CE108.aspx

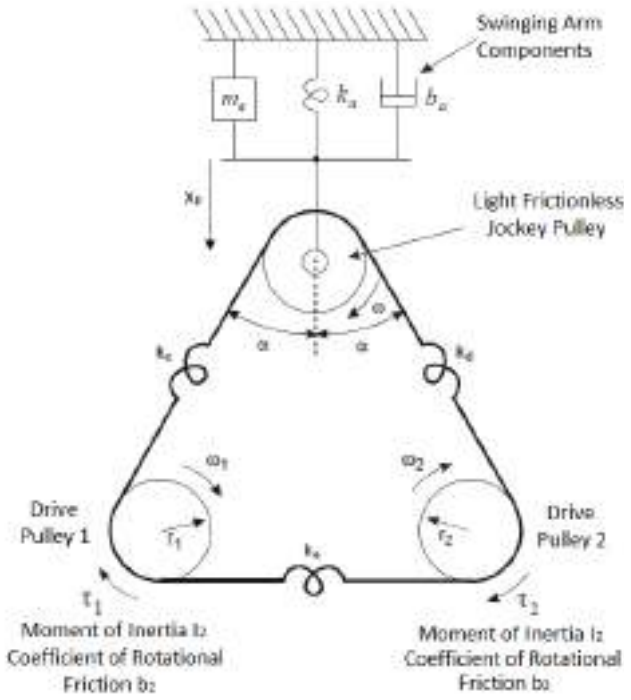

FIGURE 1. Equivalent dynamic components of coupled electric drives [32].

By using this equipment, students can learn techniques for speed and tension control, simultaneous control of velocity and tension, and analysis of multivariable control systems.

The mathematical model for the coupled electric drives system is well known [32] and can be obtained using the scheme in Figure 1. Using the lagrangian mechanics, the kinetic energy, and the dissipation term of motors and pulley, it follows that:

$$\mathrm{T} = \frac{1}{2}m\dot{x}_p^2 + \frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}I_2\dot{\theta}_2^2 + \frac{1}{2}I_p\dot{\theta}_p^2 \quad (1)$$

$$\mathrm{R} = \frac{1}{2}b_1\dot{\theta}_1^2 + \frac{1}{2}b_2\dot{\theta}_2^2 + \frac{1}{2}b_{pa}\dot{\theta}_p^2 + \frac{1}{2}b_{pt}\dot{x}_p^2 \quad (2)$$

Where x is the jockey pulley vertical position, $[I_1, I_2, I_p]$ are the motors and pulley inertia, m is the jockey pulley mass, $[b_1, b_2]$ are the motors friction, $[b_{pt}, b_{pa}]$ are the translational and angular pulley friction, and $[\theta_1, \theta_2, \theta_p]$ are the angular positions.

The elastic nature of the belt can be approximated to a spring so that the potential energy is:

$$\mathrm{V} = \frac{1}{2}[r(\theta_1 - \theta_2)]^2 + \frac{1}{2}k[r(\theta_1 - \theta_p) - x\cos(\alpha)]^2 + \frac{1}{2}k[r(\theta_p - \theta_2) - x\cos(\alpha)]^2 + \frac{1}{2}k_0x^2 \quad (3)$$

Where $[k, k_0]$ are the belt and spring stiffness and r is the radius of the pulley and the motors, which are assumed to be equal. Neglecting the pulley inertia and angular dissipation and using equations (1), (2) and (3), we finally obtain the following expressions where the torques τ_1 and τ_2 are the system inputs:

$$\mathrm{I}_1\ddot{\theta}_1 + b_1\dot{\theta}_1 + kr^2[\frac{3}{2}\theta_1 - \frac{3}{2}\theta_2 - \frac{x}{r}\cos(\alpha)] = \tau_1 \quad (4)$$

$$\mathrm{I}_2\ddot{\theta}_2 + b_2\dot{\theta}_2 + kr^2[-\frac{3}{2}\theta_1 + \frac{3}{2}\theta_2 + \frac{x}{r}\cos(\alpha)] = \tau_2 \quad (5)$$

$$m\ddot{x} + b_{pt}\dot{x} - kr\theta_1\cos(\alpha) + xk_0 + kr[\theta_1\cos(\alpha) + \frac{2x}{r}\cos^2(\alpha)] = 0 \quad (6)$$

TABLE 1. Coupled electric drives parameters.

m	r	I	b	k	k_0	b_{pt}	α
0.35	0.03	$8\cdot10^{-4}$	$9\cdot10^{-2}$	50	200	0.5	$\pi/6$
kg	m	kgm^2	Nm/s	N/m	N/m	N/s	

The equations (4), (5) and (6) are used in the virtual laboratory, taking $I = I_1 = I_2 = I_p$, $b = b_1 = b_2$ and using the values for the parameters of the coupled drives system in Table 1. Dynamic equations in matrix notation could be written as:

$$M = \begin{pmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & m \end{pmatrix}'' \quad B = \begin{pmatrix} b & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & b_{pt} \end{pmatrix}$$

$$K = \begin{pmatrix} \frac{3}{2}kr^2 & -\frac{3}{2}kr^2 & -kr\cos(\alpha) \\ -\frac{3}{2}kr^2 & \frac{3}{2}kr^2 & kr\cos(\alpha) \\ -kr\cos(\alpha) & kr\cos(\alpha) & k_0 + 2k\cos^2(\alpha) \end{pmatrix}$$

$$M \cdot \frac{d^2}{dt^2}\begin{pmatrix} \theta_1 \\ \theta_1 \\ x \end{pmatrix} + B \cdot \frac{d}{dt}\begin{pmatrix} \theta_1 \\ \theta_1 \\ x \end{pmatrix} + K = \begin{pmatrix} \tau_1 \\ \tau_2 \\ 0 \end{pmatrix}$$

Finally, with the Laplace transform and rewriting the output and input vector to match with the inputs and outputs of the system:

$$Z(s) = G(s) \cdot U(s)$$

$$U(s) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$Z(s) = \begin{pmatrix} s/2 & s/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \theta_1 \\ \theta_1 \\ x \end{pmatrix}$$

Then, the transfer functions of this MIMO system are:

$$\begin{pmatrix} \dot{\theta}_p \\ x \end{pmatrix} = \begin{pmatrix} \omega_p \\ x \end{pmatrix} = \begin{pmatrix} G_\omega & G_\omega \\ -G_x & G_x \end{pmatrix}\begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix}$$

$$\mathrm{G}_\omega(s) = \frac{0.5}{Is + b} \quad (7)$$

$$\mathrm{G}_x(s) = \frac{4600}{(s^2 + 1.6s + 790)(s^2 + 111s + 1)} \quad (8)$$

Equation (7) (the belt's velocity) matches with a first order system, while in equation (8) (the belt's tension) we get a transfer function with two real and two complex poles.

B. THE VIRTUAL LABORATORY

EJS allows introducing the differential equations of a model (in this case, the model of the coupled electric drives system detailed in the previous section) to create an interactive simulation with a graphical user interface (GUI). This GUI consists of buttons, sliders, numerical fields, check boxes, graphs, and two and three dimensions graphical elements that allow changing and visualizing parameters of the lab.

Figure 2 shows the basic structure of the virtual laboratory. The simulation window is divided into three sections: a menu, the system 2D graphical representation and control, and the evolution graphs and indicators.

- Menu: In the top left corner of the VL's user interface there are four buttons:
  - *Files:* Used to save four different graphs: the belt's position, velocity, and tension, and the motors' angles. From this menu, users can also save the numeric data in a .m file for later analysis with Matlab.
  - *Control:* This menu button allows modifying the mode in which the experiment runs: in a manual way (open loop) or with an automatic control (closed loop with a controller).
  - *Language:* To select between english and spanish.
  - *Step Time:* This menu contains four different step times for user purposes. For example, the analysis of the velocity transient needs a low step time like $dt = 0.0001$ seconds.
- System representation and control: In last section Figure 1 shows a 2D representation of the system (two motors at the bottom, a flexible belt, and a jockey pulley at the top), that its included in the laboratory application as shown in Figure 2. Under this visualization zone the application shows four tabs:
  - In the *Controls* tab (Figure 2.a), students can change the velocity of the motors using the U_1 and U_2 sliders or introducing the values directly in the editable numeric fields at the right. In automatic mode, the U_1 and U_2 sliders are disabled and the user can change the tension and velocity references (the two inputs of this system), and select whether to make a coupled or a decoupled control for these variables.
  - The *PID* tab (Figures 2.b and 2.c) shows three or six fields to adjust the PID parameters of tension and/or velocity controllers, depending on the kind of control that is selected: coupled or decoupled. In coupled mode (Figure 2.b), students have to choose the variable they want to control, either the tension or the velocity of the belt. In decoupled mode (Figure 2.c), both PID controllers are activated and the user can change their six parameters.
  - The *Controller* tab (Figure 2.d) contains a text field where advanced users can write their own controller. This code is interpreted in the control loop and, when the user clicks in the *"use the code"* checkbox, it replaces the built-in PID used in this simulation. The controller code can be loaded or saved using the respective buttons. A window containing a list of the variables she might use for programming the controller is displayed when clicking in the *"In/Out Help"* checkbox.
  - The *AC Signals* tab (Figure 2.e) provides the possibility of studying the system's time response by introducing a sinusoidal input signal. A first slider allows changing the velocity of the simulation while a second one can be used to change the estimation period for the gain and phase showed in the *"Ac Signals"* graph. In addition, selecting the *"Auto-Bode"* checkbox, new options appear in this tab (Figure 2.f), to allow configuring an automatic *"ac sweep"*.

  Finally, the VL has three more buttons (in its bottom-left part) for controlling the execution of the simulation: *play*, *pause* and *reset*.
- Evolution graphs and indicators: The right part of the virtual lab application offers a graphical representation of the most important variables in different tabs: position, velocity, and tension of the belt, motor/pulley angles, auxiliar variables (when a custom controller, and not the built-in PID, is being used) and estimation of gain and phase.

C. THE REMOTE LABORATORY

The architecture of previous remote labs in UNED were usually based on three software tools: EJS, LabVIEW (Laboratory Virtual Instrument engineering Workbench) and the JIL server [33]–[36]. With this approach, an EJS application implements the user interface in the client side, allowing students to interact with the plant, carry out experiments and obtain experimental data. In the server PC, there is a LabVIEW VI (Virtual Instrument) which is plant-dependent and implements a local control and some safety measures to avoid damaging the plant in case of accident or malicious users. This server PC needs to be running the JIL server, which offers an interface for the EJS application in the client side to interact with the LabVIEW VI. This architecture has worked very well over the last years, and nowadays there are many RLs in use based on this solution. The main successful idea of *JIL Server* was the introduction of a middleware built over the local control software, that adds an interoperability layer, and reduces the coupling between the client and the server implementation.

In this line, the effort of the present work focuses in the generalization of the described architecture. This one is chosen based on the experience gained with *JIL Server* over the last years, and the believe that the use of open standards and technologies can help to reduce the economical cost and efforts required for buildinging RLs. Therefore, based on the methods provided by *JIL Server* [37], authors define

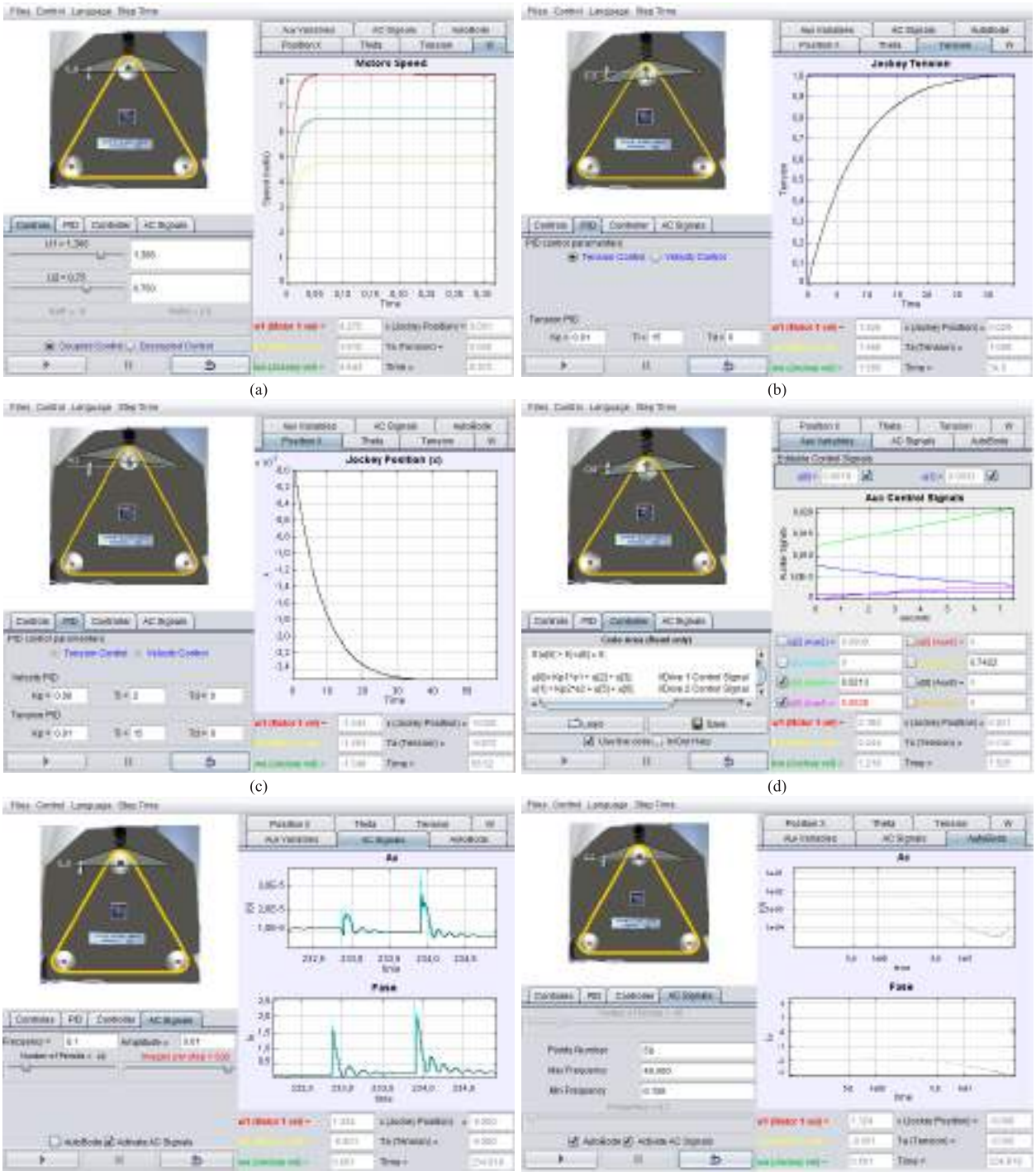

FIGURE 2. Different views of the virtual laboratory application of the coupled electric drives. (a) Controls tab and plot of the angular velocity for a manual control. (b) PID tab for a coupled automatic tension control and the output. (c) PID tab for an decoupled automatic tension and velocity control and plot of the jockey position. (d) Controller tab, with a controller programmed in java code in the text area. The evolution window contains the plot and values of the input/output variables that can be used in the code. (e) AC Signals tabs. The one on the left allows configuring the sinusoidal input signal, while the one on the right shows the magnitude and phase estimation. (f) AC Signals tab in *AutoBode* mode. The one on the left shows the configuration of a frequency sweep of fifty points. The one on the right offers the graphs of the magnitude and phase values gathered in the sweep.

a new interoperability API (Application Program Interface) to expose services to the client: the *Remote Hardware Interoperability (RHI) Protocol*. The RHI protocol provides a well defined interface to get state measurement and control and handle user requests. The interoperability API relies on two standard *remote procedure calling* protocols (RPC):

XML-RPC and *JSON-RPC*. This is somewhat similar to the *smart device paradigm* [38], which aims at enhancing the software while keeping the same hardware.

In summary, as a general approach, RLs development can be decomposed into the middleware tier that implements the RHI protocol, and the hardware control, which usually must perform several tasks: *Communication*, *Data-Acquisition and Control*, and *Data Logging*.

In the remote lab presented in this paper, the server PC, connected locally with the plant, has been replaced by a low-cost development platform: the *BeagleBone Black* (BBB) board. The BBB is a low-cost, community-supported development platform for developers and hobbyists. This board has many interesting I/O capabilities, such as *USB* client and host, *Ethernet*, *HDMI*, *GPIO* with *PWM* and even built-in support for *I2C* and *SPI*. It is shipped with a pre-installed *Angstrom Linux* distro, which is optimized for embedded systems, but it is compatible with several Linux distros, such as *Debian* or *Ubuntu*, and even with *Android*.

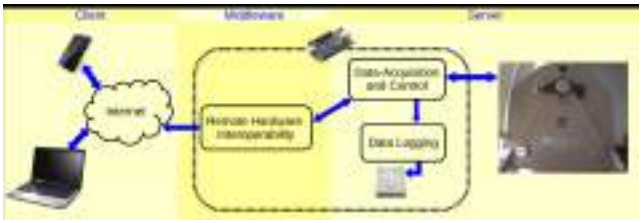

FIGURE 3. The architecture of the remote lab.

The architecture looks rather similar to the previous labs presented in [33]–[36] (see Figure 3), but the *LabVIEW VI* and *JIL server* functions have been assumed by *Node.js*, a lightweight and efficient platform which is built on *Chrome's JavaScript runtime*. This platform has been chosen because it is a ready-to-use tool, built-in in the installation of any BBB, and, as its website claims, it is for "easily building fast, scalable network applications". The client interface for the remote lab is still implemented as an EJS application, similar to the virtual laboratory described in the previous Section (see Figure 4). Therefore the thoroughly change of the server is transparent to the user. As seen in Figure 4, the GUI for the RL is almost identical to the VL one. Therefore, no further explanation is required in this regard.

The motivation for changing the PC by the BBB is twofold. On the one hand, it is much cheaper and it can, at least for this purpose, do the same function. On the other hand, the equipment used in the laboratory requires space, and PCs are much more bulky. Though it is possible to replicate the previous communication architecture based on LabVIEW with a BeagleBone Black board, there are at least two reasons that encourage looking for an alternative:

- *Economical cost*. Since BBB is chosen because it is a low cost and open hardware platform, it would be contradictory to use a software with an expensive commercial license such as LabVIEW is.

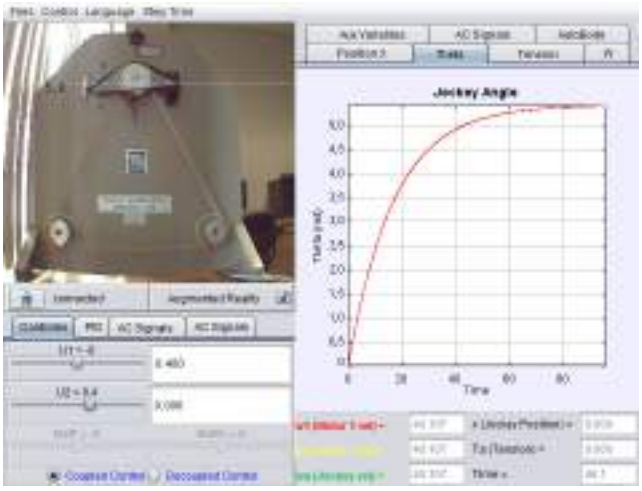

FIGURE 4. The remote lab application.

- *Computational cost*. Though the board is built over an *ARM Cortex A8* processor, which might be able to run LabVIEW, it seems oversized for the application. It is more reasonable to use a less resource-consuming software.

The middleware and the server tiers are implemented in the BBB. The framework is divided into three subsystems: *Communication*, *Data-Acquisition and Control*, and *Data Logging*.

The *communication* subsystem provides an implementation of the RHI protocol. Since there is a current trend to propose client in Javascript for compatibility with tablets, smartphones and similar devices, the use of *JSON-RPC*[10] becomes interesting due to its simplicity: it is human-readable (and, therefore, easier to debug), and the use of the *JavaScript Object Notation (JSON)* facilitates the integration with javascript applications.

The *data-acquisition and control* subsystem directly interfaces with the plant hardware, reading the velocities from the motors and the tension from the belt, and sending the control actions (received from the user interaction with the EJS application in the client side) to the motors.

Finally, the *data-logging subsystem* consists of a low priority loop which writes to disk the system events (sensor readings, user commands, etc). Though the data-logging for the user is done in EJS and integrated with UNILabs, this is also useful for debugging in case there are any malfunctioning problems.

With respect to the connection between the plant and the board, the signal measured from the sensors and the control voltages admitted by the motors are within the range $(-15V, 15V)$. However, the analogs inputs of the BBB board admit only a range of $(0V, 1.8V)$, while the outputs only can provide values in the range $(0V, 3.3V)$. So to adapt the levels, a *signal-conditioning* block was designed (basically

[10]http://www.jsonrpc.org/specification